加微信：642945106 发送“ 赠送 ”领取赠送精品课程
发数字“ 2 ”获取众筹列表

08 | 服务发现：到底是要CP还是AP?

2020-03-06 何小锋

RPC实战与核心原理 进入课程 >

讲述：张浩
时长 13:31 大小 10.84M

你好，我是何小锋。在上一讲中，我讲了“怎么设计一个灵活的 RPC 框架”，总结起来，就是怎么在 RPC 框架中应用插件，用插件方式构造一个基于微内核的 RPC 框架，其关键点就是“插件化”。

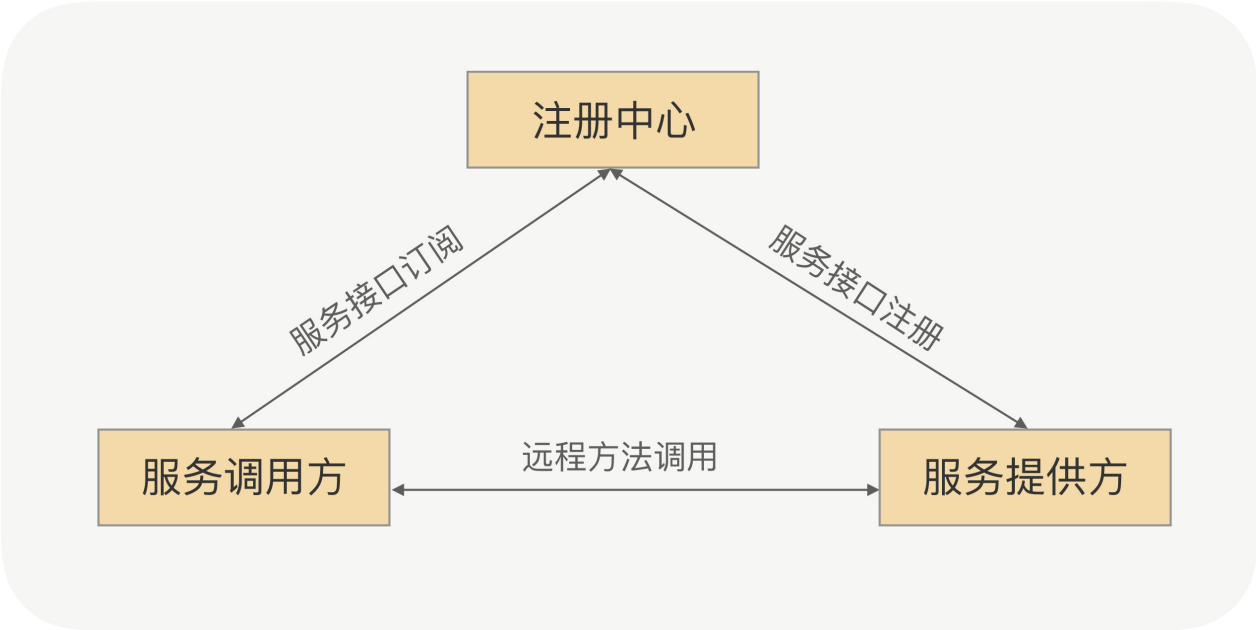
今天，我要和你聊聊 RPC 里面的“服务发现”在超大规模集群的场景下所面临的挑战。

为什么需要服务发现?

先举个例子，假如你要给一位以前从未合作过的同事发邮件请求帮助，但你却没有他的邮箱地址。这个时候你会怎么办呢？如果是我，我会选择去看公司的企业“通信录”。

同理，为了高可用，在生产环境中服务提供方都是以集群的方式对外提供服务，集群里面的这些 IP 随时可能变化，我们也需要用一本“通信录”及时获取到对应的服务节点，这个获取的过程我们一般叫作“服务发现”。

对于服务调用方和服务提供方来说，其契约就是接口，相当于“通信录”中的姓名，服务节点就是提供该契约的一个具体实例。服务 IP 集合作为“通信录”中的地址，从而可以通过接口获取服务 IP 的集合来完成服务的发现。这就是我要说的 PRC 框架的服务发现机制，如下图所示：



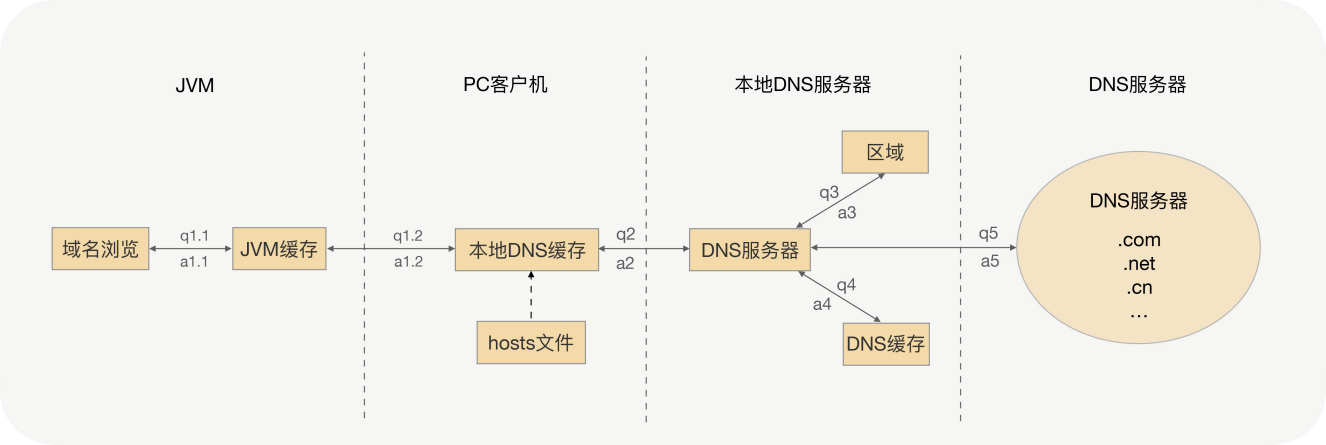
RPC服务发现原理图

1. 服务注册：在服务提供方启动的时候，将对外暴露的接口注册到注册中心之中，注册中心将这个服务节点的 IP 和接口保存下来。
2. 服务订阅：在服务调用方启动的时候，去注册中心查找并订阅服务提供方的 IP，然后缓存到本地，并用于后续的远程调用。

为什么不使用 DNS?

既然服务发现这么“厉害”，那是不是很难实现啊？其实类似机制一直在我们身边，我们回想下服务发现的本质，就是完成了接口跟服务提供者 IP 的映射。那我们能不能把服务提供者 IP 统一换成一个域名啊，利用已经成熟的 DNS 机制来实现？

好，先带着这个问题，简单地看下 DNS 的流程：



DNS查询流程

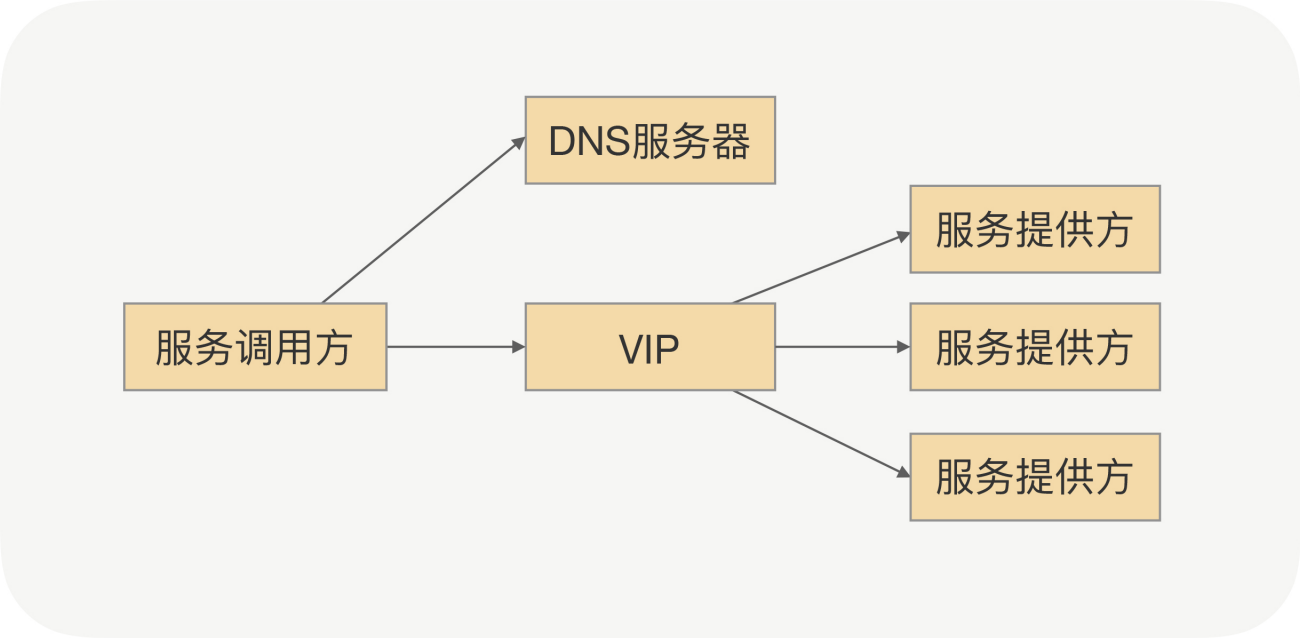
如果我们用 DNS 来实现服务发现，所有的服务提供者节点都配置在了同一个域名下，调用方的确可以通过 DNS 拿到随机的一个服务提供者的 IP，并与之建立长连接，这看上去并没有太大问题，但在我们业界为什么很少用到这种方案呢？不知道你想过这个问题没有，如果没有，现在可以停下来想想这样两个问题：

> 如果这个 IP 端口下线了，服务调用者能否及时摘除服务节点呢？
>
> 如果在之前已经上线了一部分服务节点，这时我突然对这个服务进行扩容，那么新上线的服务节点能否及时接收到流量呢？

仿冒必究 请务必加
官方微信：17161143665

这两个问题的答案都是：“不能”。这是因为为了提升性能和减少 DNS 服务的压力，DNS 采取了多级缓存机制，一般配置的缓存时间较长，特别是 JVM 的默认缓存是永久有效的，所以说服务调用者不能及时感知到服务节点的变化。

这时你可能会想，我是不是可以加一个负载均衡设备呢？将域名绑定到这台负载均衡设备上，通过 DNS 拿到负载均衡的 IP。这样服务调用的时候，服务调用方就可以直接跟 VIP 建立连接，然后由 VIP 机器完成 TCP 转发，如下图所示：



VIP方案

这个方案确实能解决 DNS 遇到的一些问题，但在 RPC 场景里面也并不是很合适，原因有以下几点：

搭建负载均衡设备或 TCP/IP 四层代理，需求额外成本；

请求流量都经过负载均衡设备，多经过一次网络传输，会额外浪费些性能；

负载均衡添加节点和摘除节点，一般都要手动添加，当大批量扩容和下线时，会有大量的人工操作和生效延迟；

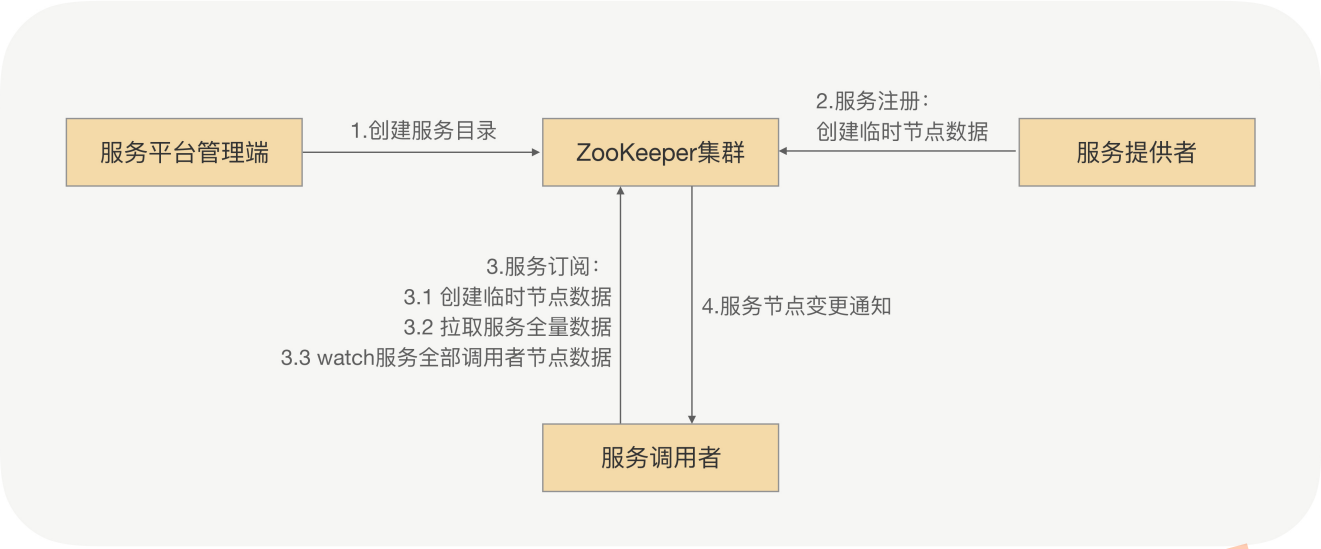
我们在服务治理的时候，需要更灵活的负载均衡策略，目前的负载均衡设备的算法还满足不了灵活的需求。

由此可见，DNS 或者 VIP 方案虽然可以充当服务发现的角色，但在 RPC 场景里面直接用还是很难的。

基于 ZooKeeper 的服务发现

那么在 RPC 里面我们该如何实现呢？我们还是要回到服务发现的本质，就是完成接口跟服务提供者 IP 之间的映射。这个映射是不是就是一种命名服务？当然，我们还希望注册中心能完成实时变更推送，是不是像开源的 ZooKeeper、etcd 就可以实现？我很肯定地说“确实可以”。下面我就来介绍下一种基于 ZooKeeper 的服务发现方式。

整体的思路很简单，就是搭建一个 ZooKeeper 集群作为注册中心集群，服务注册的时候只需要服务节点向 ZooKeeper 节点写入注册信息即可，利用 ZooKeeper 的 Watcher 机制完成服务订阅与服务下发功能，整体流程如下图：



基于ZooKeeper服务发现结构图

1. 服务平台管理端先在 ZooKeeper 中创建一个服务根路径，可以根据接口名命名（例如：/service/com.demo.xxService），在这个路径再创建服务提供方目录与服务调用方目录（例如：provider、consumer），分别用来存储服务提供方的节点信息和服务调用方的节点信息。
2. 当服务提供方发起注册时，会在服务提供方目录中创建一个临时节点，节点中存储该服务提供方的注册信息。
3. 当服务调用方发起订阅时，则在服务调用方目录中创建一个临时节点，节点中存储该服务调用方的信息，同时服务调用方 watch 该服务的服务提供方目录（/service/com.demo.xxService/provider）中所有的服务节点数据。
4. 当服务提供方目录下有节点数据发生变更时，ZooKeeper 就会通知给发起订阅的服务调用方。

我所在的技术团队早期使用的 RPC 框架服务发现就是基于 ZooKeeper 实现的，并且还平稳运行了一年多，但后续团队的微服务化程度越来越高之后，ZooKeeper 集群整体压力也越来越高，尤其在集中上线的时候越发明显。“集中爆发”是在一次大规模上线的时候，当时有超大批量的服务节点在同时发起注册操作，ZooKeeper 集群的 CPU 突然飙升，导致 ZooKeeper 集群不能工作了，而且我们当时也无法立马将 ZooKeeper 集群重新启动，一直到 ZooKeeper 集群恢复后业务才能继续上线。

拼课微信：17116143665

经过我们的排查，引发这次问题的根本原因就是 ZooKeeper 本身的性能问题，当连接到 ZooKeeper 的节点数量特别多，对 ZooKeeper 读写特别频繁，且 ZooKeeper 存储的目录达到一定数量的时候，ZooKeeper 将不再稳定，CPU 持续升高，最终宕机。而宕机之后，由于各业务的节点还在持续发送读写请求，刚一启动，ZooKeeper 就因无法承受瞬间的读写压力，马上宕机。

这次“意外”让我们意识到，ZooKeeper 集群性能显然已经无法支撑我们现有规模的服务集群了，我们需要重新考虑服务发现方案。

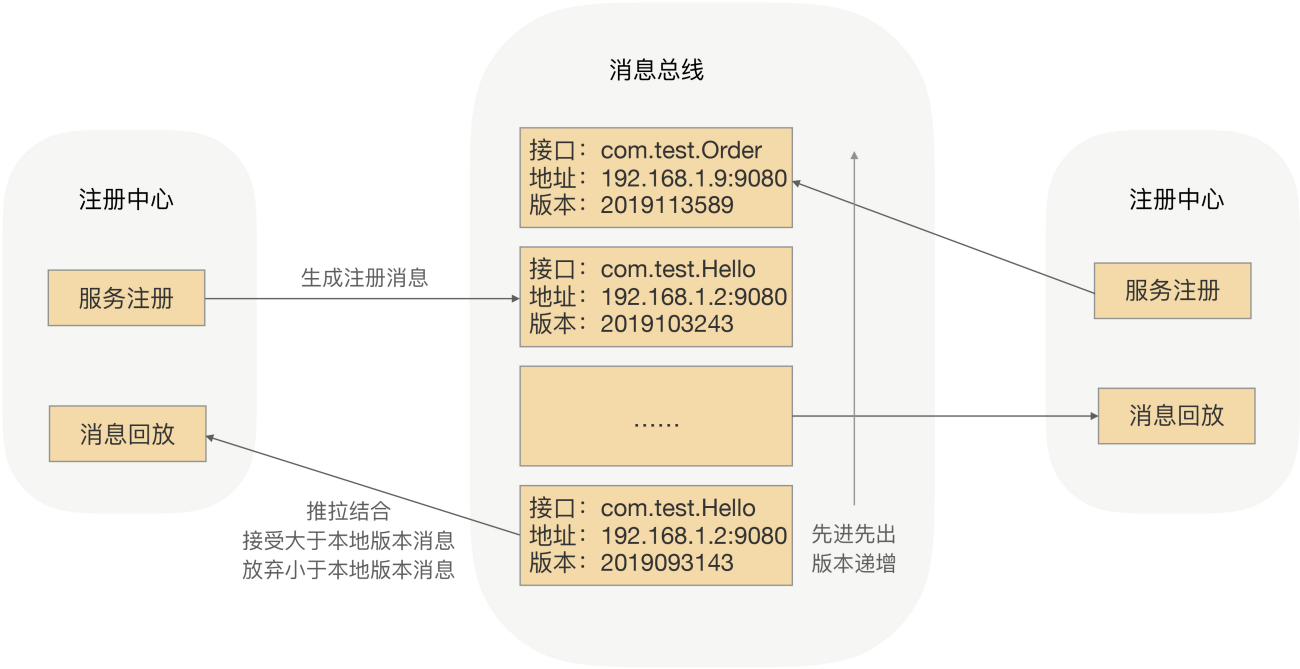
基于消息总线的最终一致性的注册中心

我们知道，ZooKeeper 的一大特点就是强一致性，ZooKeeper 集群的每个节点的数据每次发生更新操作，都会通知其它 ZooKeeper 节点同时执行更新。它要求保证每个节点的数据能够实时的完全一致，这也就直接导致了 ZooKeeper 集群性能上的下降。这就好比几个人在玩传递东西的游戏，必须这一轮每个人都拿到东西之后，所有的人才能开始下一轮，而不是说我只要获得到东西之后，就可以直接进行下一轮了。

而 RPC 框架的服务发现，在服务节点刚上线时，服务调用方是可以容忍在一段时间之后（比如几秒钟之后）发现这个新上线的节点的。毕竟服务节点刚上线之后的几秒内，甚至更长的一段时间内没有接收到请求流量，对整个服务集群是没有什么影响的，所以我们可以牺牲掉 CP（强制一致性），而选择 AP（最终一致），来换取整个注册中心集群的性能和稳定性。

那么是否有一种简单、高效，并且最终一致的更新机制，能代替 ZooKeeper 那种数据强一致的数据更新机制呢？

因为要求最终一致性，我们可以考虑采用消息总线机制。注册数据可以全量缓存在每个注册中心内存中，通过消息总线来同步数据。当有一个注册中心节点接收到服务节点注册时，会产生一个消息推送给消息总线，再通过消息总线通知给其它注册中心节点更新数据并进行服务下发，从而达到注册中心间数据最终一致性，具体流程如下图所示：



流程图

当有服务上线，注册中心节点收到注册请求，服务列表数据发生变化，会生成一个消息，推送给消息总线，每个消息都有整体递增的版本。

消息总线会主动推送消息到各个注册中心，同时注册中心也会定时拉取消息。对于获取到消息的在消息回放模块里面回放，只接受大于本地版本号的消息，小于本地版本号的消息直接丢弃，从而实现最终一致性。

消费者订阅可以从注册中心内存拿到指定接口的全部服务实例，并缓存到消费者的内存里面。

采用推拉模式，消费者可以及时地拿到服务实例增量变化情况，并和内存中的缓存数据进行合并。

为了性能，这里采用了两级缓存，注册中心和消费者的内存缓存，通过异步推拉模式来确保最终一致性。

另外，你也可能会想到，服务调用方拿到的服务节点不是最新的，所以目标节点存在已经下线或不提供指定接口服务的情况，这个时候有没有问题？这个问题我们放到了 RPC 框架里面去处理，在服务调用方发送请求到目标节点后，目标节点会进行合法性验证，如果指定接口服务不存在或正在下线，则会拒绝该请求。服务调用方收到拒绝异常后，会安全重试到其它节点。

通过消息总线的方式，我们就可以完成注册中心集群间数据变更的通知，保证数据的最终一致性，并能及时地触发注册中心的服务下发操作。在 RPC 领域精耕细作后，你会发现，服务发现的特性是允许我们在设计超大规模集群服务发现系统的时候，舍弃强一致性，更多地考虑系统的健壮性。最终一致性才是分布式系统设计中更为常用的策略。

总结

今天我分享了 RPC 框架服务发现机制，以及如何用 ZooKeeper 完成“服务发现”，还有 ZooKeeper 在超大规模集群下作为注册中心所存在的问题。

通常我们可以使用 ZooKeeper、etcd 或者分布式缓存（如 Hazelcast）来解决事件通知问题，但当集群达到一定规模之后，依赖的 ZooKeeper 集群、etcd 集群可能就不稳定了，无法满足我们的需求。

在超大规模的服务集群下，注册中心所面临的挑战就是超大批量服务节点同时上下线，注册中心集群接受到大量服务变更请求，集群间各节点间需要同步大量服务节点数据，最终导致如下问题：

注册中心负载过高；

各节点数据不一致；

服务下发不及时或下发错误的服务节点列表。

RPC 框架依赖的注册中心的服务数据的一致性其实并不需要满足 CP，只要满足 AP 即可。我们就是采用“消息总线”的通知机制，来保证注册中心数据的最终一致性，来解决这些问题的。

另外，在今天的内容中，很多知识点不只可以应用到 RPC 框架的“服务发现”中。例如服务节点数据的推送采用增量更新的方式，这种方式提高了注册中心“服务下发”的效率，而这种方式，你还可以利用在其它地方，比如统一配置中心，用此方式可以提升统一配置中心下发配置的效率。

课后思考

目前服务提供者上线后会自动注册到注册中心，服务调用方会自动感知到新增的实例，并且流量会很快打到该新增的实例。如果我想把某些服务提供者实例的流量切走，除了下线实

例，你有没有想到其它更便捷的办法呢？

欢迎留言和我分享你的思考和疑惑，也欢迎你把文章分享给你的朋友，邀请他加入学习。我们下节课再见！

更多课程推荐

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇　07 | 架构设计：设计一个灵活的RPC框架

下一篇　09 | 健康检测：这个节点都挂了，为啥还要疯狂发请求？

精选留言 (19)

写留言

每天晒白牙

2020-03-06

今天这个思考题好像是在说负载均衡策略，那是不是可以加个权重，把想下线的ip权重置为0，这样服务调用方就不会调用这个节点了

展开

3

君言
2020-03-06

老师，在AP实现中“两级缓存，注册中心和消费者的内存缓存，通过异步推拉模式来确保最终一致性”能展开讲一下具体实现吗？另外请教下CP可以基于zk实现，AP在业内的实现方式有哪些呢？

作者回复: 推主要实现callback，拉的动作在客户端，像Eurek属于AP

2

嬰
2020-03-07

如果要能切换流量，那么要服务端配置有权重负载均衡策略，这样服务器端可以通过调整权重来安排流量

作者回复: 很好的思路

1

松花皮蛋me
2020-03-06

路由负载

展开

作者回复: 很棒

1

阿卧
2020-03-06

zookeeper注册中心实现原理
1. 服务平台向zookeeper创建服务目录
2. 服务提供者向zookeeper创建临时节点
3. 服务调用者订阅zookeeper，创建临时节点，拉取服务全量数据，watch服务全部节点数据...

展开

作者回复: 这还是利用了服务发现

1

楼下小黑哥
2020-03-06

服务消费者都是从注册中心拉取服务提供者的地址信息，所以我们要切走某些服务提供者数据，只需要将注册中心这些实例的地址信息删除（其实下线应用实例，实际也是去删除注册中心地址信息），然后注册中心反向通知消费者，消费者受到拉取最新提供者地址信息就没有这些实例了。
老师，提问一个问题：现有开源注册中心是不是还没有消息总线这种实现方式？消息总...
展开

作者回复: 通过服务发现来摘除流量是最常见的手段，还可以上下线状态、权重等方式。现成的MQ也是可以充当消息总线来用

1

成熟中的猪
2020-03-09

如文中描述： ZooKeeper 的节点数量特别多，对 ZooKeeper 读写特别频繁，且 ZooKeeper 存储的目录达到一定数量的时候，ZooKeeper 将不再稳定，CPU 持续升高，最终宕机。
这里有点笼统，是否有量化的压测数据或者实际经验值可以分享？
展开

忆澜
2020-03-09

老师，求解答.....我看到这个思考题，第一反应是负载均衡，我看评论很多也是，也得到了老师的肯定，由此我有几个问题
1.在这种rpc框架中，是不是客户端和注册中心都会有负载均衡？（如果是的话，那这道题答案应该是客户端的吧）如果都有，那他们各自职责怎么明确呢，感觉总会有一些交集？
2.如果客户端有负载均衡，那这些配置也是存在注册中心上么？如果是的话，比如像题中...
展开

作者回复: 配置信息由客户端配置，也可由配置中心管理

wusiration

2020-03-08

负载均衡，将对应节点的流量切为0

展开

作者回复: 很棒

Reason

2020-03-08

要把服务流量切走还可以采用客户端服务负载均衡和服务路由的方式。服务路由则采取面向标签的路由方式，根据标签值确定选择访问的服务端实例

展开

作者回复: 很棒

问心

2020-03-08

老师，DNS+负载均衡的灵活性差体现在哪呢。现在负载均衡都提供了接口去注册，配置负载百分比，同时还可以使用http、tcp、udp的健康检查用于自动上下线。是有哪些高级玩法，这样不能用么？

作者回复: 这样负载均衡自身就有瓶颈，而且很难满足个性需求，比如：分组、路由等

小美

2020-03-08

请问老师，dubbo的服务发现也是用zk实现的，在大规模时有这个问题呢？

作者回复: 是的

shen

2020-03-08

直接再管理后台将注册中心的路由下线有可以了

展开

作者回复: 操作有点复杂

Pick Monster ...

2020-03-07

消息总栈类似一个队列，队列表示是递增的数字，注册中心集群的任何一个节点接收到注册请求，都会把服务提供者信息发给消息总栈，消息总栈会像队列以先进先出的原则推送消息给所有注册中心集群节点，集群节点接收到消息后会比较自己内存中的当前版本，保存版本大的，这种方式有很强的实效性，注册中心集群也可以从消息总栈拉取消息，确保数据AP，个人理解这是为了防止消息未接收到导致个别节点数据不准确，因为服务提供...

展开

作者回复: 上面理解的很棒。event bus可以改造成主从模式保证高可用

(Kelen)

2020-03-07

终于看到这个消息总线的另一个应用，我们的配置中心也利用了这个原理

作者回复: 很棒

郝志强

2020-03-07

想到通过修改负载均衡算法的负载权重，一般通过配置中心修改

作者回复: 很棒

小鱼儿

2020-03-07

老师，你好。能够通过nginx进行负载均衡，consul进行服务注册与发现。服务提供方通过注册器注册到consul中，consul模板更新nginx。而调用发只需要连nginx即可。

作者回复: nginx的存在就变成集中式了

hello
2020-03-06

一般会提供一个后台API接口，将需要下线的实例先从注册中心移除。期待老师巧妙地思路！！！

作者回复: 可以考虑权重配置

lingjiango
2020-03-06

流量网关?

展开